ILOG CPLEX 8.1

Release Notes

December 2002

© Copyright 2001, 2002 by ILOG

This document and the software described in this document are the property of ILOG and are protected as ILOG trade secrets. They are furnished under a license or non-disclosure agreement, and may be used or copied only within the terms of such license or non-disclosure agreement.
No part of this work may be reproduced or disseminated in any form or by any means, without the prior written permission of ILOG S.A.

Printed in France

Table of Contents

Conversion Notes 5
New Features 5
Concert Technology Version 6
Advanced Documentation and Examples 6
Additional information. 6
Release Notes for CPLEX 8.0 (reproduced). 7
Conversion Notes for CPLEX 8.0. 7
Solution Status Codes 7
Problem Types. 14
Other Changes Affecting All Users 14
Other Changes Affecting Component Library Users 16
New Features in CPLEX 8.0 17
Algorithmic Features 17
Component Libraries Usage 20

ILOG CPLEX 8.1 — Release Notes

Please review these brief release notes carefully before using ILOG CPLEX 8.1. They are organized in three sections:

- Conversion Notes—a guide to the upgrade procedure from ILOG CPLEX 8.0 to ILOG CPLEX 8.1.
- New Features—an overview of the features introduced in ILOG CPLEX 8.1.
- Additional information.

Conversion Notes

No conversion of application source code is required when moving from CPLEX version 8.0 to version 8.1. CPLEX 8.0 license keys can be used with version 8.1.

For details on conversion from CPLEX 7.5 to CPLEX 8.0, please refer to *Conversion Notes for CPLEX 8.0*.

New Features

This section lists the new features in ILOG CPLEX 8.1.

Concert Technology Version

CPLEX 8.1 uses ILOG Concert Technology 1.3 for compatibility with ILOG Solver 5.3.

Advanced Documentation and Examples

The *ILOG CPLEX 8.1 Advanced Reference Manual* is included with the standard documentation set.

Advanced examples are available in `cplex81/examples/src/`.

Additional information

If you are upgrading from ILOG CPLEX 8.0, these notes should be all you need. If you are upgrading from a version of CPLEX prior to 8.0, please refer first to the *Release Notes for CPLEX 8.0 (reproduced)* included for your convenience in this document, and to any prior release notes (ILOG CPLEX 7.5, ILOG CPLEX 7.1, and so forth) to bring your ILOG CPLEX software up-to-date from your particular version.

The descriptions in these release notes are summaries; full descriptions of the features mentioned will be found in:

- *ILOG CPLEX 8.1 Getting Started;*
- *ILOG CPLEX 8.1 User's Manual;*
- *ILOG CPLEX 8.1 Reference Manual;*
- *ILOG CPLEX 8.1 Java Reference Manual;* and
- *ILOG CPLEX 8.1 Advanced Reference Manual.*

All these manuals are provided in the distributed files, in various online forms. *Getting Started* and the *User's Manual* are available in printed form at the version 8.0 level.

Release Notes for CPLEX 8.0 (REPRODUCED)

Please review these release notes carefully before using ILOG CPLEX 8.0. They are organized into two sections:

- Conversion Notes — a guide to the upgrade procedure from ILOG CPLEX 7.5 to ILOG CPLEX 8.0, and
- New Features — an overview of the features introduced in ILOG CPLEX 8.0.

If you are upgrading from ILOG CPLEX 7.5, then these notes should be all you need. If you are upgrading from a version of CPLEX prior to 7.5, please refer first to the *ILOG CPLEX 7.5 Release Notes*, as well as any prior release notes (ILOG CPLEX 7.1, ILOG CPLEX 7.0, and so forth) to bring your ILOG CPLEX software up-to-date from your particular version.

Conversion Notes for CPLEX 8.0

These Conversion Notes are intended to help you convert from ILOG CPLEX 7.5 to ILOG CPLEX 8.0 as quickly and easily as possible. We have divided the notes into four sections. The first describes changes to solution status codes. These changes will affect most users, so we recommend that everyone reads this section. The second section describes changes to problem types that will also affect many users. The third section describes other changes affecting all users. The fourth section describes additional changes that affect users who embed ILOG CPLEX optimizers in their applications.

Solution Status Codes

Solution status codes for LP and QP have been completely revised in ILOG CPLEX 8.0, making the interface simpler and more unified. This section describes the changes that must be made to existing applications to work with the new status codes. The new codes are described in more detail in Appendix B of the *ILOG CPLEX 8.0 Reference Manual*.

MIP status codes have been changed too, but the changes are much smaller in scope than those for LP and QP. The MIP changes are discussed in *MIP Solution Status Codes* below.

The major change in status codes is that their meaning no longer depends on the optimizer used; solution status codes now relate to the problem being solved. This section includes Tables 1 to 6 below that show how to convert from the previous optimizer-specific interpretations to the new problem interpretations.

Another change is that the optimizers will no longer return an error value when either presolve or consistency checks have found the problem to be infeasible or unbounded. These conditions now result in a normal termination. Note that a zero return from an optimizer or a nonzero return from `CPXgetstat()` no longer means that a solution exists and can be queried, as it did in previous versions of ILOG CPLEX. For non-optimal solutions, one must check for the existence of a solution with the new solution information routines, described

below. The handling of these conditions by Concert Technology has not changed; an exception is thrown when a solution query is made, and there is no solution.

We have also changed the meaning of the unbounded return code slightly. ILOG CPLEX 8.0 will declare a problem to be unbounded when the problem has an unbounded ray. In previous versions, an unbounded status also implied the existence of a feasible solution. The new solution information routines can be used to determine whether a feasible solution was found, as described below. See the note in Appendix B of the *ILOG CPLEX 8.0 Reference Manual* for more information on the meaning of the unbounded status.

Information about the status of an aborted optimization (that is, feasibility of the current solution) was previously encoded in the solution status. This information must now be queried separately, through new solution information routines described below.

Solution Status Codes for Callable Library Users

The following tables provide a mapping from status codes in ILOG CPLEX 7.5 to the new status codes in ILOG CPLEX 8.0. Note that the mapping depends on the specific algorithm used in ILOG CPLEX 7.5, so be sure to consult the appropriate table.

Table 1 *Conversion of* `CPXgetstat()` *return value when* `CPXgetmethod()` *is* `CPX_ALG_PRIMAL`

ILOG CPLEX 7.5 and earlier		ILOG CPLEX 8.0	
Symbolic Constant	**Value**	**Symbolic Constant**	**Value**
CPX_OPTIMAL	1	CPX_STAT_OPTIMAL	1
CPX_INFEASIBLE	2	CPX_STAT_INFEASIBLE	3
CPX_UNBOUNDED	3	CPX_STAT_UNBOUNDED	2
CPX_OBJ_LIM	4	CPX_STAT_ABORT_OBJ_LIM	12
CPX_IT_LIM_FEAS	5	CPX_STAT_ABORT_IT_LIM	10
CPX_IT_LIM_INFEAS	6	CPX_STAT_ABORT_IT_LIM	10
CPX_TIME_LIM_FEAS	7	CPX_STAT_ABORT_TIME_LIM	11
CPX_TIME_LIM_INFEAS	8	CPX_STAT_ABORT_TIME_LIM	11
CPX_NUM_BEST_FEAS	9	CPX_STAT_NUM_BEST	6
CPX_NUM_BEST_INFEAS	10	CPX_STAT_NUM_BEST	6
CPX_OPTIMAL_INFEAS	11	CPX_STAT_OPTIMAL_INFEAS	5
CPX_ABORT_FEAS	12	CPX_STAT_ABORT_USER	13

Table 1 *Conversion of* `CPXgetstat()` *return value when* `CPXgetmethod()` *is* `CPX_ALG_PRIMAL`

ILOG CPLEX 7.5 and earlier		ILOG CPLEX 8.0	
Symbolic Constant	**Value**	**Symbolic Constant**	**Value**
CPX_ABORT_INFEAS	13	CPX_STAT_ABORT_USER	13
CPX_ABORT_CROSSOVER	18	CPX_STAT_ABORT_USER	13

Table 2 *Conversion of* `CPXgetstat()` *return value when* `CPXgetmethod()` *is* `CPX_ALG_DUAL`

ILOG CPLEX 7.5 and earlier		ILOG CPLEX 8.0	
Symbolic Constant	**Value**	**Symbolic Constant**	**Value**
CPX_OPTIMAL	1	CPX_STAT_OPTIMAL	1
CPX_INFEASIBLE	2	CPX_STAT_UNBOUNDED	2
CPX_UNBOUNDED	3	CPX_STAT_INFEASIBLE	3
CPX_OBJ_LIM	4	CPX_STAT_ABORT_OBJ_LIM	12
CPX_IT_LIM_FEAS	5	CPX_STAT_ABORT_IT_LIM	10
CPX_IT_LIM_INFEAS	6	CPX_STAT_ABORT_IT_LIM	10
CPX_TIME_LIM_FEAS	7	CPX_STAT_ABORT_TIME_LIM	11
CPX_TIME_LIM_INFEAS	8	CPX_STAT_ABORT_TIME_LIM	11
CPX_NUM_BEST_FEAS	9	CPX_STAT_NUM_BEST	6
CPX_NUM_BEST_INFEAS	10	CPX_STAT_NUM_BEST	6
CPX_OPTIMAL_INFEAS	11	CPX_STAT_OPTIMAL_INFEAS	5
CPX_ABORT_FEAS	12	CPX_STAT_ABORT_USER	13
CPX_ABORT_INFEAS	13	CPX_STAT_ABORT_USER	13
CPX_ABORT_CROSSOVER	18	CPX_STAT_ABORT_USER	13

Table 3 *Conversion of* `CPXgetstat()` *return value when* `CPXgetmethod()` *is* `CPX_ALG_BARRIER`

ILOG CPLEX 7.5 and earlier		ILOG CPLEX 8.0	
Symbolic Constant	**Value**	**Symbolic Constant**	**Value**
CPX_OPTIMAL	1	CPX_STAT_OPTIMAL	1
CPX_IT_LIM_FEAS	5	CPX_STAT_ABORT_IT_LIM	10
CPX_IT_LIM_INFEAS	6	CPX_STAT_ABORT_IT_LIM	10
CPX_TIME_LIM_FEAS	7	CPX_STAT_ABORT_TIME_LIM	11
CPX_TIME_LIM_INFEAS	8	CPX_STAT_ABORT_TIME_LIM	11
CPX_OPTIMAL_INFEAS	11	CPX_STAT_OPTIMAL_INFEAS	5
CPX_ABORT_DUAL_INFEAS	14	CPX_STAT_ABORT_USER	13
CPX_ABORT_PRIM_INFEAS	15	CPX_STAT_ABORT_USER	13
CPX_ABORT_PRIM_DUAL_INFEAS	16	CPX_STAT_ABORT_USER	13
CPX_ABORT_PRIM_DUAL_FEAS	17	CPX_STAT_ABORT_USER	13
CPX_PRIM_INFEAS	32	CPX_STAT_INFEASBLE	3
CPX_DUAL_INFEAS	33	CPX_STAT_UNBOUNDED	2
CPX_PRIM_DUAL_INFEAS	34	CPX_STAT_INFEASIBLE	2
CPX_PRIM_OBJ_LIM	35	CPX_STAT_ABORT_PRIM_OBJ_LIM	21
CPX_DUAL_OBJ_LIM	36	CPX_STAT_ABORT_DUAL_OBJ_LIM	22
CPX_OPTIMAL_FACE_UNBOUNDED	37	CPX_STAT_OPTIMAL_FACE_UNBOUNDED	20
CPX_NUM_BEST_PRIM_DUAL_FEAS	38	CPX_STAT_NUM_BEST	6
CPX_NUM_BEST_PRIM_INFEAS	39	CPX_STAT_NUM_BEST	6
CPX_NUM_BEST_DUAL_INFEAS	40	CPX_STAT_NUM_BEST	6
CPX_NUM_BEST_PRIM_DUAL_INFEAS	41	CPX_STAT_NUM_BEST	6
CPX_BARRIER_NUM_ERROR	42	CPX_STAT_NUM_BEST	6
CPX_BARRIER_INCONSISTENT	43	no longer used	

Table 4 *Conversion of error return values from any of the optimizers*

ILOG CPLEX 7.5 and earlier		ILOG CPLEX 8.0	
Symbolic Constant	**Value**	**Symbolic Constant**	**Value**
CPXERR_BOUNDS_INFEAS	1100	CPX_STAT_INFEASIBLE	3
CPXERR_PRESLV_INForUNBD	1101	CPX_STAT_INForUNBD	4
CPXERR_PRESLV_INF	1117	CPX_STAT_INFEASIBLE	3
CPXERR_PRESLV_UNBD	1118	CPX_STAT_UNBOUNDED	2
CPXERR_DEMAND_BALANCE	1533	CPX_STAT_INFEASIBLE	3

Solution Status Codes for Concert Technology Users

Table 5 contains the new `IloCplex::getCplexStatus()` return values when the model has been solved with the primal simplex or dual simplex algorithm (that is, `IloCplex::getAlgorithm()` returns `IloCplex::Primal` or IloCplex::Dual).

Table 5 *Conversion of IloCplex::getCplexStatus() — Primal or Dual Simplex*

ILOG CPLEX 7.5 and earlier	ILOG CPLEX 8.0
Optimal	Optimal
Infeasible	Infeasible
Unbounded	Unbounded
InforUnbd	InforUnbd
ObjLim	AbortObjLim
ItLimFeas	AbortItLim
ItLimInfeas	AbortItLim
TimeLimFeas	AbortTimeLim
TimeLimInfeas	AbortTimeLim
NumBestFeas	NumBest
NumBestInfeas	NumBest
OptimalInfeas	OptimalInfeas

Table 5 *Conversion of IloCplex::getCplexStatus() — Primal or Dual Simplex*

ILOG CPLEX 7.5 and earlier	ILOG CPLEX 8.0
`AbortFeas`	`AbortUser`
`AbortInfeas`	`AbortUser`
`AbortCrossover`	`AbortUser`

Table 6 contains the new `IloCplex::getCplexStatus()` return values when the model has been solved with the barrier algorithm (that is, `IloCplex::getAlgorithm()` returns `IloCplex::Barrier`).

Table 6 *Conversion of IloCplex::getCplexStatus() — Barrier*

ILOG CPLEX 7.5 and earlier	ILOG CPLEX 8.0
`Optimal`	`Optimal`
`InforUnbd`	`InforUnbd`
`ItLimFeas`	`AbortItLim`
`ItLimInfeas`	`AbortItLim`
`TimeLimFeas`	`AbortTimeLim`
`TimeLimInfeas`	`AbortTimeLim`
`OptimalInfeas`	`OptimalInfeas`
`AbortDualInfeas`	`AbortUser`
`AbortPrimInfeas`	`AbortUser`
`AbortPrimDualInfeas`	`AbortUser`
`AbortPrimDualFeas`	`AbortUser`
`PrimInfeas`	`Infeasible`
`DualInfeas`	`Unbounded`
`PrimDualInfeas`	`Infeasible`
`PrimObjLim`	`AbortPrimObjLim`
`DualObjLim`	`AbortDualObjLim`
`OptimalFaceUnbounded`	`OptimalFaceUnbounded`
`NumBestPrimDualFeas`	`NumBest`

Table 6 *Conversion of IloCplex::getCplexStatus() — Barrier*

ILOG CPLEX 7.5 and earlier	ILOG CPLEX 8.0
NumBestPrimnfeas	NumBest
NumBestDualInfeas	NumBest
NumBestPrimDualInfeas	NumBest
BarrierNumError	NumBest
BarrierInconsistent	no longer used

New Solution Query Routines

The new Callable Library functions `CPXsolninfo()` and `CPXNETsolninfo()` are provided as a means to access solution information produced by LP and QP optimization routines. Some of this information was previously encoded in the solution status values. In Concert Technology (for C++ and Java users) this functionality is provided with methods `IloCplex::isPrimalFeasible()` and `IloCplex::isDualFeasible()`. The functions are described in the *ILOG CPLEX 8.0 Reference Manual* and their usage is explained in the *ILOG CPLEX 8.0 User's Manual*.

If an `Infeasible` or `Unbounded` code is returned, there may not be a solution available. Check for the existence of a solution with the `solntype` parameter `CPXsolninfo()`, for example:

```
status = CPXsolninfo (env, lp, &solntype, &method,
                      &pfeasind, &dfeasind);
```

If the `solntype` is `CPX_NO_SOLN`, then there is no solution, and any of the solution query routines, for example, `CPXgetitcnt()`, `CPXsolution()`, will result in an error. When a solution is available, the `solntype` value indicates whether the solution is basic (value `CPX_BASIC_SOLN`) or not (`CPX_NONBASIC_SOLN`).

As mentioned above, Concert Technology users will get an exception if a solution is queried but no solution is available. This behavior is the same as in previous versions.

The solution information functions also provide a simple means to find out whether the solution is primal or dual feasible (through the `pfeasind` and `dfeasind` arguments of routine `CPXsolninfo()` in the Callable Library, or the `isPrimalFeasible()` and `isDualFeasible()` methods of class `IloCplex` in Concert Technology). One use of this is to determine if a primal feasible solution exists when the solution status is `Unbounded`.

MIP Solution Status Codes

The only solution status change for MIP is the addition of two new status codes: `CPXMIP_UNBOUNDED` and `CPXMIP_INForUNBD`. ILOG CPLEX 7.5 would return an error code if it found a MIP model to be infeasible or unbounded.

Note that in ILOG CPLEX 7.5, a model would be declared infeasible if the branch & cut optimizer found no feasible solutions, while in ILOG CPLEX 8.0 this model will be declared infeasible or unbounded. The new status code more accurately describes the conclusion that can be reached from this outcome. The user may turn off dual reductions, using the presolve reduction parameter (`CPX_PARAM_REDUCE` or `IloCplex::Reduce`), if a firm infeasible conclusion is desired.

Problem Types

To accommodate the new MIQP problem class, ILOG CPLEX problem types have been reorganized and streamlined. The allowed values for problem type in the C interface are now:

- `CPXPROB_LP`;
- `CPXPROB_MILP`;
- `CPXPROB_FIXEDMILP`;
- `CPXPROB_QP`;
- `CPXPROB_MIQP`; and
- `CPXPROB_FIXEDMIQP`.

Consult the *ILOG CPLEX 8.0 Reference Manual* entry `CPXchgprobtype()` for details. Concert Technology now interprets all problems as MIQPs and treats MILP, QP, and LP as special cases.

The problem types `CPXPROB_RELAXED` (now `CPXPROB_RELAXEDMILP`) and `CPXPROB_ZEROEDQP` (and their counterparts `solveRelaxed` and `solveZeroedQP` in C++ and Java, or `relaxed_mip` and `zeroed_qp` in the Interactive Optimizer), will be removed in the next release. We recommend you now convert any procedures with these problem types to use, typically, problem type `CPXPROB_LP` in Callable Library programs.

Concert Technology users should use `IloConversion` objects instead of using `solveRelaxed()`, and should replace a quadratic objective with a linear objective (`model.remove(quadraticObj); model.add(linearObj);`) instead of using `solveZeroedQP()`. Removal of other problem types is not contemplated at this point. Contact ILOG Technical Support if you need assistance in identifying the appropriate changes.

Other Changes Affecting All Users

Parallel algorithms are now included in the standard ILOG CPLEX binaries and libraries. Parallelism is controlled through user parameters, in conjunction with limits set by your ILOG CPLEX license key. Most users of the ILOG CPLEX Technologies (Interactive Optimizer and the various Component Libraries) will now have just one version to be concerned with, instead of distinct parallel and serial versions. The exception is users

interested in the parallel dual simplex optimizer: that will be included in a separate OpenMP® binary.

The default parallel thread count has been changed to one. It previously defaulted to its maximum possible value, as determined by the number of processors in the machine and the number of threads licensed.

The behavior of the default `Automatic` setting of `CPX_PARAM_LPMETHOD` has been changed. This setting will now choose Primal Simplex in cases where a primal-feasible starting basis is available. Also, if the number of parallel threads (global default thread count) is set to a value greater than `1`, and an advanced simplex basis is not available, the new `Concurrentopt` feature (see *Concurrentopt* on page 19) will be invoked. Concert Technology will give this same behavior for the default `Automatic` settings of parameters `IloCplex::RootAlg` (C++) or `IloCplex.IntParam.RootAlg` (Java).

The MIP backtrack parameter (`CPX_PARAM_BTTOL` or `IloCplex::Bttol`) controls the amount of backtracking in the branch & cut search; its default value has been changed, from `0.01` to `0.9999`, because of various changes made in the overall branching strategy. As a conversion issue, this change would affect only users who for some reason manually set ILOG CPLEX parameters to their default values. Most users will find the new default value works well.

Users may no longer choose “Dual to iteration limit then barrier” or “Barrier without crossover” as root or node algorithms in MIP. Use the barrier crossover parameter (`BarCrossAlg` in Concert Technology, `CPX_PARAM_BARCROSSALG` in the Callable Library and `BARRIER CROSSOVER` in the Interactive Optimizer) if you wish to use barrier without crossover.

The default value for the `BarGrowth` parameter (`CPX_PARAM_BARGROWTH` or `IloCplex::BarGrowth`) has been increased from 10^6 to 10^8 to reduce the likelihood that a model will be declared to have an unbounded optimal face.

The barrier optimizer no longer has an explicit iteration count limit by default. In other words, `BarItLim`, the barrier iteration limit parameter (`CPX_PARAM_BARITLIM` or `IloCplex::BarItLim`) now defaults to a very large value.

The method for expressing the quadratic term of the objective function in ILOG CPLEX LP format for file input has changed. The factor of `0.5` will no longer be assumed for the terms appearing within square brackets. The factor is specified by appending the characters `'/ 2'` after the closing bracket. In order to ease conversion, the implied factor will continue to be assumed for this release; however, a warning message will be printed that says

```
Warning: Line 234: deprecated format, expecting '] / 2'.
```

We recommend you begin using this new form for expressing the quadratic term of a QP now.

While error code CPXERR_PRESLV_INForUNBD (Error 1101) is no longer returned by the optimizers, as noted in the previous section on status code changes, it can still arise when a user attempts to write the presolved model to a file via the routine CPXpreslvwrite().

SGI users need IRIX 6.5.9 or above to run ILOG CPLEX 8.0.

Other Changes Affecting Component Library Users

User callback functions written in C must be re-defined to use const for certain parameters. The function definition must use the const modifier, and the pointer argument must be declared as type CPXCENVptr (the constant variant of CPXENVptr). Moreover, the function itself must not change any const arguments or pass them to functions where they are not const; that is, const arguments cannot be changed within a user-written callback.

Several features that were previously announced as obsolete have now been removed. They are the CPLEX License Manager, the Old Memory Model and several symbolic constant names appearing in C include file cplex.h. The Kernighan & Ritchie style C function prototypes also have been removed. Consult earlier CPLEX Release Notes documents if you still need to perform one of these conversions.

The IloCplex member functions setRootAlgorithm, setNodeAlgorithm, getRootAlgorithm, and getNodeAlgorithm, have been replaced by parameters and are now deprecated features which will be removed in a future release. For example, setRootAlgorithm(alg) becomes setParam(IloCplex::RootAlg, alg). See the discussion related to these in “New Features in CPLEX 8.0” on page 17 of these Release Notes.

To support a unified approach to LP and QP models, function IloCplex::LPCallbackI has been renamed to IloCplex::ContinuousCallbackI.

Also, the callbacks IloCplex::PrimalSimplexCallbackI, and IloCplex::DualSimplexCallbackI have been combined into a single callback IloCplex::SimplexCallbackI.

IloCplex::CallbackI::makeClone() has been renamed to IloCplex::CallbackI::duplicateCallback(). Programs that use the macros for defining callbacks will be unaffected by this change.

IloCplex::Algorithm::NetworkPrimal and IloAlgorithm::NetworkDual have been merged into IloAlgorithm::Network.

Routine CPXcangetbase() has been removed. Identical functionality is now available through the new CPXsolninfo() routine.

To improve clarity and consistency, the names of a few Concert Technology for C++ methods and objects have been changed:

- Method `isQP()` is now `isQuadratic()`.
- Methods `getLb()`, `getLbs()`, `getUb()` and `getUbs()` have been changed to `getLB()`, `getLBs()`, `getUB()`, and `getUBs()`.
- The return value of functions `getCplexStatus()` and `getCplexSubStatus()` has been changed from `Status` to `CplexStatus`.
- The functions for querying basis information have been renamed to `getBasisStatus()` and `getBasisStatuses()` according to Table 7 *New Names for Basis Query Functions* on page 17.

In all cases, the old names are still supported, but will be discontinued in the next release.

Table 7 *New Names for Basis Query Functions*

Old	New
`BasisStatus getStatus(const IloNumVar con) const;`	`BasisStatus getBasisStatus(const IloNumVar con) const;`
`BasisStatus getStatus(const IloRange con) const;`	`BasisStatus getBasisStatus(const IloRange con) const;`
`void getStatuses(BasisStatusArray stat, const IloNumVarArray con) const;`	`void getBasisStatuses(BasisStatusArray stat, const IloNumVarArray con) const;`
`void getStatuses(BasisStatusArray stat, const IloRangeArray con) const;`	`void getBasisStatuses(BasisStatusArray stat, const IloRangeArray con) const;`
`void getStatuses(BasisStatusArray cstat, const IloNumVarArray var, BasisStatusArray rstat, const IloRangeArray con) const;`	`void getBasisStatuses(BasisStatusArray cstat, const IloNumVarArray var, BasisStatusArray rstat, const IloRangeArray con) const;`

New Features in CPLEX 8.0

Here are the new features found in the Application Programming Interface (API) that you may find useful in solving particular kinds of models.

ILOG CPLEX 8.0 contains many performance enhancements that benefit all problem types without any need to change or add to your present solution procedures. But in addition, various features have been added that provide new functionality. Here is an outline of what you may wish to try. Refer to the *ILOG CPLEX 8.0 User's Manual* and *ILOG CPLEX 8.0 Reference Manual* for complete details on these features.

Algorithmic Features

The algorithmic features added in this version are:

- Goals
- Simplex optimizers for QP problems
- Mixed Integer QP problems
- Sifting
- Concurrentopt for multiple processor computers
- MIP parameters
- Barrier parameters
- Function to remove an SOS from the problem

Goals

Goals, which have proven to be quite powerful and widely used in Constraint Programming, are now available for ILOG CPLEX MIP. Goals provide a new level of control for users who want to manage the way that ILOG CPLEX explores the MIP search tree. See the *ILOG CPLEX 8.0 User's Manual* for full details on using goals in ILOG CPLEX.

Simplex Optimizers for Quadratic Programming Problems

Dual Simplex and Primal Simplex optimizers have been added for solving Quadratic Programming (QP) problems; in prior ILOG CPLEX releases QP problems were solved only by the Barrier optimizer, which can not restart an optimization from a previous solution. Additionally, Presolve has been extended to perform new reductions on QP problems, benefitting all the QP optimizers. See the *ILOG CPLEX 8.0 User's Manual* for full details on these new capabilities for solving QP problems.

Mixed Integer Quadratic Programming Problems

The branch & cut algorithm has been extended to permit discrete variables in QP formulations, so that Mixed Integer Quadratic Programming (MIQP) is now supported. That is, a problem of the new type `CPXPROB_MIQP` may contain any combination of Continuous, Binary, General Integer, Semi-Continuous, or Special Ordered Set variables, in conjunction with an objective function that may contain a quadratic term.

See the *ILOG CPLEX 8.0 User's Manual* for full details on these new capabilities for solving MIQP problems.

Sifting

A new method, Sifting, is now provided for solving continuous LP models. It is activated by setting the parameter `CPX_PARAM_LPMETHOD` to the value `CPX_ALG_SIFTING(5)`, and calling `CPXlpopt()`. For Concert, it is activated by setting parameter `IloCplex::RootAlg` to `IloCplex::Sifting` (C++), or `IloCplex.IntParam.RootAlg` to `IloCplex.Algorithm.Sifting` (Java), and invoking method `cplex.solve()`. Sifting solves a sequence of LP subproblems, where the results from one subproblem are used to select columns from the original model for inclusion in the next subproblem. This iterative

sifting process eventually converges to an optimal solution for the original model. Sifting is especially applicable to models with many more columns than rows.

Concurrentopt

A new method is provided for solving LP models on multiple-CPU computers (with appropriate ILOG CPLEX licensing). If the `CPX_PARAM_LPMETHOD` parameter is set to `CPX_ALG_AUTOMATIC(0)` and no advanced simplex basis is loaded, or if the parameter is set to `CPX_ALG_CONCURRENT(6)`, subsequent `CPXlpopt()` calls will launch LP optimizations on multiple threads at once. For Concert Technology users, this capability is exercised by setting `IloCplex::RootAlg` to `IloCplex::AutoAlg` or `IloCplex::Concurrent` (C++), or by setting `IloCplex.IntParam.RootAlg` to `IloCplex.Algorithm.Auto` or `IloCplex.Algorithm.Concurrent` (Java), and invoking method `cplex.solve()`.

The first thread uses Dual Simplex, if a second thread is available it uses Barrier (if your license supports Barrier), and the next thread uses Primal Simplex. If additional threads are available on the computer, they are applied to the Barrier run by an invocation of Parallel Barrier. Unfortunately, this does not happen on SGI platforms due to a problem in the threading libraries. On a machine with enough memory, this will result in a solution being returned by the fastest of the available algorithms on each problem, eliminating the need to choose a single optimizer for all purposes.

MIP Parameters

The default MIP emphasis setting (`CPX_PARAM_MIPEMPHASIS` or `IloCplex::MipEmphasis`) has been changed to a new value of `Balanced` (`CPX_MIPEMPHASIS_BALANCED` or `MIPEmphasisBalanced`), which seeks a balance between searching for feasible solutions and proving optimality. There should be no conversion issue related to this change, because the previous `Optimality` default has been renumbered to `2`, and the new default remains numbered `0`. The new `BestBound` choice (`CPX_MIPEMPHASIS_BESTBOUND` or `MIPEmphasisBestBound`) emphasizes moving the Best Bound as an aggressive technique for proving optimality on extremely difficult models.

The other choice for this parameter, `CPX_MIPEMPHASIS_FEASIBILITY`, is unchanged. Most models will reach the optimal solution fastest using the `Optimality` emphasis, but the default `Balanced` setting produces good feasible solutions earlier and more frequently, while only slightly degrading time to proven optimality.

The Concert Technology enumerations for parameters `IloCplex::MIPEmphasis` (C++) and `IloCplex.IntParam.MIPEmphasis` (Java) have been changed accordingly. Refer to the ILOG CPLEX include files for details.

The branch & cut search can be fine-tuned with the new dive type parameter (`CPX_PARAM_DIVETYPE` or `IloCplex::DiveType`). This parameter controls how much look-ahead, or probing, is done while processing nodes. The default value is the automatic setting (`0`) which will do some probing. The setting of `1` activates the strategy used in previous ILOG CPLEX versions, and the setting of `2` does probing all the time.

The new parameter CPX_PARAM_SYMMETRY controls symmetry-breaking cuts (`IloCplex::Symmetry` (C++) and `IloCplex.IntParam.Symmetry` (Java) for Concert users), a Preprocessing feature. These cuts are off by default, but turning them on can be helpful on some models.

Barrier Parameters

The feasibility algorithms for Barrier (`CPX_PARAM_BARALG` settings `1` and `2`) now support solution of QP models.

Function to Remove an SOS

`CPXdelsetsos()` is a new function which removes one or more special ordered sets (SOS) from the problem.

Component Libraries Usage

The use of `const` in C function prototypes allows the specification of arguments that will not be changed by execution of a function. This modifier has been added throughout `cplex.h`. Several new `typedefs` have been added to ease the use of `const` with ILOG CPLEX structure types, including `CPXCENVptr`, `CPXCLPptr` and `CPXCNETptr`. This addition will not affect existing programs except ones that use callback functions, as mentioned in the Conversion Notes section of this document.

Two new functions are provided to work with ILOG CPLEX parameters in C:

- `CPXgetparamname()` which provides the string corresponding to a parameter number, and
- `CPXgetparamnum()` which provides the number corresponding to a string matching a parameter name.

A new parameter `CPX_PARAM_QPMETHOD` and new routine `CPXqpopt()` (Callable Library only), are available to control the choice of optimizer for the solution of QP models. This same functionality is already provided in Concert through the `solve()` function, which now no longer issues a warning message if a Simplex optimizer is selected for QP.

A new parameter `CPX_PARAM_QPMAKEPSDIND` (`IloCplex::QPmakePSDInd` for Concert Technology users) determines whether ILOG CPLEX will attempt to adjust a binary MIQP formulation so that its quadratic matrix is positive semi-definite (PSD).

A new function `CPXqpindefcertificate()` (`IloCplex::qpIndefCertificate()` for Concert Technology users) is available. In cases where the quadratic matrix in the objective function of a QP model violates the assumption of positive semi-definiteness, it returns a vector that demonstrates the violation.

Two new query routines are provided to support the new sifting algorithm. These are `CPXgetsiftitcnt()` (`IloCplex::getNsiftingIterations()` (C++) and `IloCplex.getNsiftingIterations` (Java) for Concert Technology users), and `CPXgetsiftphase1cnt()` (`IloCplex::getNsiftingPhaseOneIterations()`

(C++) and `IloCplex.getNsiftingPhaseOneIterations()` (Java) for Concert users). `CPXgetsiftitcnt()` returns the number of sifting iterations performed during the most recent optimization, and `CPXgetsiftphase1cnt()` returns the number of sifting phase 1 iterations performed by the most recent optimization before a feasible solution was found.

C++ parameters `IloCplex::RootAlg` and `IloCplex::NodeAlg` now are available, as is the case for Java users. The parameters replace the functionality previously provided by the `IloCplex` member functions `setRootAlgorithm`, `setNodeAlgorithm`, `getRootAlgorithm`, and `getNodeAlgorithm`. These four functions are now deprecated features and will be removed in a future release.

Concert Technology users now have a function `isQuadratic()` which indicates whether the current problem contains a quadratic term in the objective.

The new C++ function `clearModel()` is available and is used to remove the contents of a model without destroying it.

Java users now can use the static method `IloCplex.putenv(String envstr)` in a manner analogous to C and C++ usage.

For compatibility with the Java version of Concert Technology for C++ users can now use the following methods:

`getNbarrierIterations()`	`getNSOSs()`
`getNNZs()`	`getNbinVars()`
`getNsemiContVars()`	`getNsemiIntVars()`
`getRangeSA()`	`getAlgorithm()`
`getSubAlgorithm()`	`getNintVars()`

Visual Basic users no longer need to declare the functions in the CPLEX DLL themselves. They can use the new `cplex.bas` include file, distributed in the `include/ilcplex` subdirectory. There are also several new examples that illustrate how the CPLEX DLL can be called from Visual Basic programs. These can be found in the `examples/vb`.

ILOG WORLDWIDE INFORMATION CENTER
TEL: +1 775 831 7744
+1 800 367 4564

- **FRANCE**

  ILOG S.A.
  Gentilly
  Tel: +33 (0)1 49 08 35 00
  E-mail: info@ilog.fr

- **GERMANY**

  ILOG Deutschland GmbH
  Bad Homburg v.d.H.
  Tel: + 49 6172 40 60 -0
  E-mail: info@ilog.de

- **JAPAN**

  ILOG CO., LTD
  Tokyo
  Tel: +81 3 5211 5770
  E-mail: info@ilog.co.jp

- **SINGAPORE**

  ILOG (S) Pte Ltd.
  Singapore
  Tel: +65 773 06 26
  E-mail: info@ilog.com.sg

- **SPAIN**

  ILOG S.A.
  Madrid
  Tel: +34 91 710 2480
  E-mail: info@ilog.es

- **UK**

  ILOG Ltd.
  Bracknell
  Tel: +44 (0) 1344 66 16 00
  E-mail: info@ilog.co.uk

- **USA / ETATS-UNIS**

  ILOG, Inc.
  Mountain View, CA
  Tel: +1 (650) 567 8000
  E-mail: info@ilog.com

Representatives and distributors in other countries

WWW.ILOG.COM

ILOG, CPLEX and the ILOG logotype are registered trademarks, and all ILOG product names are trademarks of ILOG. All other brand, product and company names are trademarks or registered trademarks of their respective holders.

20n1e06